The Journal of Academic Librarianship 44 (2018) 689–691

Contents lists available at ScienceDirect

The Journal of Academic Librarianship

journal homepage: www.elsevier.com/locate/jacalib

In-house Software Development: Considerations for Implementation

Scott Jackson, Sian Brannon*

University of North Texas, Denton, TX, United States of America

Introduction

In 2006, Chris Stearns wrote in the *Journal of Hospital Librarianship* about the expansion of information technology (IT) staffs' responsibilities to include the in-house development of software for libraries. This is as opposed to purchasing commercial products (3rd party). Though now over a decade old, the points he raises about the advantages of this approach still resonate. The evolution of information technology requires that libraries not always wait on vendors to provide solutions to expanding needs.

Methods of software acquisition

Libraries require various kinds of software – integrated library systems, computer reservation programs, office productivity applications, budgeting processes, and survey generators, to name a few. It is difficult to find an aspect of librarianship that is not impacted by computers and software systems. Traditionally, acquisition of software has occurred in three different ways.

First, and perhaps most common, libraries can purchase commercial software from vendors. There are a plethora of vendors and developers that create software, and the vast majority of those developers are focused primarily on supporting the needs of businesses. This need not imply that the world of libraries is completely different from that of the private sector. However, for the most part, profit, tax depreciation, advertising, and other business constructs are not a primary variable in libraries. Libraries are concerned with the community, while most private firms are interested in the bottom line, and they are not bound by government restrictions and inflexibility regarding purchasing. Furthermore, vendors may include multiple modules to help businesses with various aspects of operation. Due to this, there is often a large portion of business-centered software that is not needed, adding unnecessary complexity to software purchasing decisions when considered in light of library requirements.

Next, one can find software that was created by someone else and made available openly on a source such as GitHub (https://github.com/). Rather than starting from scratch, this can provide a jumping off point to begin in-house development. With this approach, an organization can get a head start on development by using the "recipe" that another developer has created. With open source software, the code is readily available to be changed or manipulated to meet stated needs (Corbly, 2014). Users of these software and services need to be mindful of any licenses that associated with the software.

Finally, original software can be developed completely in-house. This can be accomplished by one person, but is best pursued in collaboration with other staff and/or departments in the organization. In these cases, a problem and outcome are identified. This then goes to a developer or development team to create a software solution that works with existing infrastructure, accomplishes the goal set forth by library staff, and is adaptable as the needs of the organization change. Once the solution is complete, the organization can choose to make the solution available for other libraries with similar software needs, or those which may wish to help with future development and improvements.

What types of software could be developed in house?

There are a number of applications for in-house software development. Here is a non-comprehensive list of things to consider for customized creation:

- Travel approval
- Conference management
- Event registration
- Statistics tracking
- Invoicing/purchasing
- Inventory control for remote locations
- Inventory process APIs that overlay on ILS
- Inventory for library assets
- Survey tools
- Authentication
- Security notifications
- Digital signage
- Workstation authorization
- Workstation reservations

In-house development and implementation process

The current reality of librarianship is that almost every action or

* Corresponding author.
E-mail address: sian.brannon@unt.edu (S. Brannon).

https://doi.org/10.1016/j.acalib.2018.10.008
Received 31 October 2018; Accepted 31 October 2018
Available online 06 November 2018
0099-1333/

service involves technology. Organizations seek to create processes that allow the entire organization to interact and complete tasks regardless of physical location, and also to track and manipulate this data. Application or software development follows a basic cycle: analyze, design, code, document, operation and maintenance (Kneuper, 2017). It begins when staff identify a need or wish. This issue then goes to IT developers to discuss if a technological solution can be applied. The developer meets with library staff to brainstorm and gather information about the need, what departmental processes must be followed, and what end result is desired.

The software developer will then organize the information from the department, the data structure and needed outcomes, and generate a flowchart of a software process that fulfills the requirements. Then coding begins. An initial prototype application is developed, and functionality will be tested by the requesting team or person. Then the prototype app is demonstrated to additional staff; at this point, all relevant staff may be invited to test the functionality and provide hands-on feedback. Communication of desired changes, additions, or appearance goes back to developers, and they will work on incorporating these into the software.

The above demo-test-feedback process repeats until users are satisfied with the software. After finishing touches (such as those to the visual design) are applied, the software will be installed onto a relevant server, computer, or device, to have final live testing by developers and staff. Final adjustments are made, then official training should commence. Ideally, staff would train before the software "goes live," though this is not always possible. After the software deploys, there will be continuous monitoring for bugs, updates, changes, and new needs.

UNT Libraries "Purchasing Dashboard" – success in in-house development

The University of North Texas (UNT) Libraries have had some success in developing software in-house for library needs. One example is the "Remote Storage Database," a warehouse-style tracking system for items stored in our off-campus remote storage facility. Separate from our integrated library system, it is able to pinpoint the location of a single item within deep compact shelving for easy retrieval. Creating it in-house took staff time, but saved almost $100,000 in contrast to the cost of the vendor-supplied product.

The most long-standing product developed in-house by UNT Libraries developers is the "Purchasing Dashboard." This collaboration between every department in the library is the application that solves our needs to streamline and efficiently log all purchase requests (outside of collection materials). The Dashboard includes the steps of request, approval, purchase, receipt, and archiving of transactions. It provides a log of documentation and collocation of communication, and resulted in a reduction in unnecessary emails.

The process of Dashboard development started when the UNT Library financial officer saw inefficiencies and issues with tracking purchases in the Access database they had always used. They asked Technology and Computer Operations unit (TACO) for a conversation about what was really needed. After a conversation with the institution's purchasing and budget offices, TACO explored vendor options and evaluated in-house possibilities. Developers in TACO brainstormed, then created a mock-up of what the Dashboard would look. They presented the interface, operations, and discovery process to the financial officer and department heads. After discussion and tweaking, TACO set out to do the main coding that would allow for beta-testing. The Administrative Office conducted purchasing in the Access database and in the beta-system for a few months.

After a few more conversations with developers, end-users, and dean, the Purchasing Dashboard was considered ready to launch. TACO and the financial officer conducted extensive training first to the Administrative Office (through which they were able to create detailed instructions) and then to the rest of the library. The Dashboard went live just over five years ago. In the meantime, it has been adjusted for university accounting changes. The original coding was flexible enough to add new accounts, reroute approvals, and allow for multiple purchasers. With the hiring of a new financial officer and some turnover in TACO, it's time to get fresh eyes on the product, see if improvements can be made, or find out if vendors have improved their offerings.

Why in-house development?

Libraries and vendors have different missions. In-house IT departments, however, should have the same mission as the library overall: to help the library function most efficiently. Vendors design products to appeal to the largest demographic they can; they seek to meet the needs of many types of libraries. In-house products, however, can be designed to meet an organization's specific needs while also working natively on a library's specific infrastructure. Current trends see vendors consolidating, offering libraries fewer choices when it comes to meeting their software needs (Breeding, 2018). Pricing of vendor solutions can be a significant barrier, involving not only the initial purchase cost, but also yearly maintenance and support costs; in-house solutions can often be less expensive.

Implementation processes for in-house development differ when compared with purchasing software from a vendor. Though both begin with a desire for software that fills a need, there are marked differences in implementation time, methods of purchase of related materials, amount of training available, and how upgrades will work. With a vendor, the library may create a "request-for-proposal" or move straight into comparison of different options. There is then implementation or migration, training, and the wait for upgrades. For in-house development, there will be more work on the front end, including determining requirements, testing, and feedback, but swifter implementation and training.

When purchasing software, libraries find themselves at the mercy of vendors for upgrades and enhancements, which are often performed on the vendor's schedule, not the libraries' timeline. Perhaps the library is on an older version of a particular product – support from vendors can become less helpful as they concentrate on newer implementations. They may force a move from a product with which the library is satisfied to a different product, sometimes one more expensive or with less ideal functionality. In-house, there is straight-line access from the user to the developers, and changes can be made more swiftly. Modifications can occur more frequently or as-needed, which means faster turnaround than waiting on vendors to fix bugs.

Another consideration is the organization's data. Many marketplace vendors are moving their products to the cloud. In the agreed-upon End-User License Agreement (EULA), a great many of these vendors have full access to mine the library's data. How does this align with the library's confidentiality goals and privacy policies? What future consideration do patrons have now that some portion of the data is available to outside organizations? How is that data accessed should the company cease to exist? It can be difficult, if not impossible, to migrate to a new vendor if the organization's information disappears (Dutta, Peng, & Choudhary, 2013).

Training on internally developed software is generally hands-on, just as vendor software might be. However, the training is done alongside the person who developed the product, and thus offers more insight into how it actually works. Giving feedback and developing solutions creates staff engagement. There is a better connection with the product and service for staff who had a hand in creating the solution, rather than just talking with vendor support about how to install or fix it. The library might also gain better documentation in some cases, as the developer (who knows the local infrastructure) and staff (who experience the product firsthand) can work alongside each other to create instructions.

Why NOT in-house development?

Of course, there are various considerations for developing software in-house, and they may outweigh its benefits in some situations. Internally developed software still requires refreshing – what if the software platform becomes obsolete? All software requires re-evaluation. Is the solution from two years ago still the best option, or could it be upgraded? Are there issues that could result in security breaches? It is important to ensure that software solutions remain secure, relevant, and functional for the need they were created to address. These responsibilities must be considered when deciding on in-house versus purchased solutions.

The return on staff investment can be great, but it must be noted that software development can also be a time consuming process. It is important not to over-commit resources to creating in-house solutions; institutions must consider current staffing expertise and responsibilities, pace commitments, and maintain realistic timelines for deliverables. Before starting a development process, organizational policies regarding intellectual property and potential legal issues must be considered. Licensing the software or providing it for free on the internet may necessitate conversations with legal, technology transfer, or other departments at the institution.

One primary risk in creating software in-house is the potential loss of employees with the relevant skills to maintain it – what are the options for support, upgrade, and maintenance if the original creator no longer works for the institution? The software solution must be documented in such a way that another developer can pick up the project if required. Furthermore, good documentation is helpful even for the developer who originally wrote the software. In addition to comments within the code, documentation can quickly answer the question down the road: “Why was it done *that way*?”

Another major concern is scope creep, which must be recognized and controlled. Self-developed applications have greater potential for being more than a single solution. As applications mature, it is not uncommon to find other uses or scenarios where functionality can be added. However, a software solution's scope should remain focused and be maintainable in such a way that it does not become a run-away project. Ensure software is exceptional at a specified task or group of tasks, rather than attempting a Swiss Army Knife of everything for every task.

Do not reinvent the wheel: some business solutions are perfectly applicable to libraries. Libraries do not need a library-specific word processor; developing one would be a waste of resources when there are commercial and open-source solutions that fulfill this need and are universally accepted. Is there a solution that provides the required functionality already? If that solution provides most, but not all, of the functionality, can the product be augmented to fulfill all requirements easily and – most importantly – with less time and effort than developing a new application in-house? Focus on library needs for which there are limited vendor solutions, existing solutions are exorbitantly expensive, or where those products do not adequately meet library needs.

Conclusion

Library staff may often wish products would function differently, or have an understanding that repetitive tasks would be easier dealt with through automation. Yet these random thoughts do not always translate into workable applications because they are never pursued beyond the wishing stage. Library staff must keep IT in the loop, share the needs so that they can be addressed. IT cannot help formulate a solution if they are unaware of the problem. In-house software development probably will not be the best choice for every software application. Moreover, it is certainly not the best option for every library. Some libraries will face conflicts over IT issues with larger IT departments in the institution. Perhaps your library doesn't currently have anyone on staff with the necessary understanding of programming. On the other hand, maybe your library does have the software skills or server experience, but isn't as adept at making the software attractive to the end-user. It is important to remember, however, that it is an option, and there are cases where it is preferable. With in-house development, the library is in charge of its own software lifecycle.

In 2006, Andrew Pace said in *American Libraries* that a good future option would be a “hybrid” environment, where “vendors open up their system just a bit more so that open-source and locally developed software can play a larger part in the overall systems architecture of libraries.” Henry echoed this in 2016, saying “Working in a respectful, open, partnership-centered relationship can benefit both parties in profound ways, not the least of which is continuing to improve these products and services for the communities that rely on them.”

That has begun, through APIs and open source systems. Nevertheless, until it is common practice, libraries may still have to solve their own software needs. If your library is looking to develop current employees, or looking to hire new ones that can help with in-house development, take heed of certain skills. Be on the lookout for problem solvers – those that can see a workflow or task as it currently exists, then identify ways to make it more efficient. These staff should be concerned with constant re-evaluation of the in-house software – can it be improved? Developers would need to understand the relevant software, be aware of vendor provided options, and have the ability to collaborate and work with non-tech people.

Vendors may not be amenable to sharing with the world, but libraries can be. Philosophically, libraries may find it more advantageous to solve problems, not buy solutions. Institutions expect their students to have original ideas, think critically, and innovate. Libraries can set an example by acting similarly. Major companies are applauded for developing the next great thing – why can't libraries do the same?

References

Breeding, M. (2018). Library systems report 2018. *American Libraries, 49*(5), 22–35.
Corbly, J. (2014). The free software alternative: Freeware, open-source software, and libraries. *Information Technology and Libraries, 33*(3), 65–75.
Dutta, A., Peng, G., & Choudhary, A. (2013). Risks in enterprise cloud computing: The perspective of IT experts. *The Journal of Computer Information Systems, 53*(4), 39–48.
Henry, R. (2016). Library software vendors: Improving relationships. *Journal of Academic Librarianship, 42*(5), 620–621.
Kneuper, R. (2017). Sixty years of software development life cycle models. *IEEE Annals of the History of Computing, 39*(3), 41–54.
Pace, A. (2006). Giving homegrown software its due. *American Libraries, 10*(37), 50–51.
Stearns, C. (2006). Developing web software in-house. *Journal of Hospital Librarianship, 4*(6), 85–91.